青森公立大学経営経済学研究
第18巻第2号（2013）
Aomori Public College Journal of Management & Economics
Vol.18, No.2, 2013

Comparison of Seven Alternative Algorithms for Applied General Equilibrium Modeling

Hidetaka KAWANO※

Many new globally convergent price adjustment process have been proposed in the last 40 years. The main strength of these approaches are their globally convergent characteristics and also the possibility of their applications to a large-scale applied general equilibrium model. In order to see the empirical applicability of these approaches, in this paper I conducted seven numerical simulations of the simple illustrative model and compared their performances. The best approach was the modified Kimbell-Harrison approach which achieved a high computational precision with only 20 iterations to obtain the converged values.

1. Introduction

At the outset of its theoretical formulation in the 1950s, the major criticism of the general equilibrium framework was that it was very intractable in dealing with multi-dimensional empirical issues due to the lack of effciently operational algorithms and suffcient computational power. This criticism was signicantly weakened with the first applications of the Johansen approach in the early 1960s (Johansen[4,1960] and Dixon, Parmenter, and Powell [2, 1992]), and the Scarf approach in the late 1960s and the early 1970s (Scarf [8, 1967] and Scarf [9, 1973]). Subsequent refinements in operational algorithms have been made, while at the same time computational power has been considerably increased due to advancements in computer technology. Thus, the early criticism of general equilibrium analysis no longer holds. The number of such empirical models, called "applied general equilibrium models" (or sometimes called "computable general equilibrium model") has mushroomed in various fields of research in the last 40 years.

Many new globally convergent price adjustment processes have been proposed in the last 40 years. The main strengths of these approaches are their globally convergent characteristics and the possibility of their applications to a large-scale applied general equilibrium model. In order to see the empirical applicability of some of these approaches, in this paper I conducted numerical simulations of the simple illustrative model and compared the performances of traditional nonlinear solvers such as the Bisection method, the Secant method, and the Traub method, with four alternative approaches: the Scarf approach (Scarf [8, 1967], Scarf [9, 1973], Shoven and Whalley [11, 1992]), the Modified Tanaka-Kawano approach (Tanaka and Kawano [12, 1996]), the Modified Kimbell-Harrison approach (Kimbell-Harrison [7, 1986] and Kawano [7, 2003]), and the Joosten and

※ Aomori Public College

Talman approach [5, 1998:15-26]. The challenge for empirical application is to develop an easily implementable algorithm which is relatively fast and efficient in computing in high precision. The computational example for this paper was drawn from Washida [14, 2004: pp. 256-7] who provided the ideal structure for testing the numerical reliability of each approach. Washida [14, 2004: pp. 244-9 and pp. 256-67] also provided his program and an excellent exposition of the Scarf algorithm.

As a result of the simulation exercise, the best approach was the modified Kimbell-Harrison approach which achieved a high computational precision with only 20 iterations to obtain the converged values. I applied the method to the example drawn from Shoven and Whalley [10, 1974]. In Kawano [6, 2003: 2-27.], the number of iterations was 409 over the Scarf's algorithm, which Shoven and Whalley used to solve the same problem with 1653 iterations. The Joosten-Talman approach also achieved a high precision with just 30 iterations. The modified Tanaka-Kawano approach achieved a high precision, but after 178 iterations. The Scarf approach performed poorly and did not achieve a high precision with over 1 million iterations, even though its computational time was 0.142 seconds which is comparatively long.[1] The other approaches of the traditional non-linear solvers such as the Bisection and the Traub methods took less than 0.001 second to converge, and performed well, but the Secant performed poorly and did not achieve a high precision with five thousands iterations. These experiments were programmed in C-language, and conducted on the GCC version 4.0.1 compiler (Apple Computer, Inc.). The verified reliability of the simulation results in double precision (1.0e-15).

In section 2, the general structure of the model is specified. In section 3, the seven alternative approaches are described. In section 4, the simulation results are summarized. The conclusion follows in section 5. The complete computer output is presented in the Appendix.

[1]The same C program was run 15 years ago. It took 1.987 seconds with Intel's 333MHz Pentium II processor to run the program.

2. Model Structure

The model is a simple competitive pure exchange economy with 3 commodities indexed $j \in I^3 = \{1,2,3\}$ and two consumers indexed. $i \in I^2 = \{1,2\}$. The set of all possible price vectors is $\Re^3_+ \setminus \{0^3\}$, where $\Re^3_+ \equiv \{x \in \Re^3 \mid x_j \geq 0 \; \forall j \in I^3\}$. Each consumer $i \in I^2$ is characterized by his consumption set X^i, initial endowments ω^i, and preference relation $\preceq^i$ with the following 3 assumptions:

Assumption 1: The consumption set X^i is closed and nonempty, and it is a convex subset of $\Re^3_+$ bounded below and unbounded above, containing the set $\left\{x \in \Re^3 \mid 0 \leq x_j \leq \sum_{i=1}^{2} \omega^i_j, \; \forall j \in I^3\right\}$, where $\omega^i = (\omega^i_1, \omega^i_2, \omega^i_3)$ is the vector of initial endowments of the 3 commodities $j \in I^3$ of consumer $i \in I^2$.

Assumption 2: Each endowment of the 3 commodities $j \in I^3$ of consumer $i \in I^2$ is strictly positive, $\omega^i_j > 0, \; \forall i \in I^2, \; j \in I^3$.

Assumption 3: The preference $\preceq^i$ relation which is quasi-ordering on X^i, is continuous, weekly monotonic, and strictly convex.

Let $B^i(p) = \{x \in X^i \mid p^T x \leq p^T w^i\}$ denote the budget set of consumer $i \in I^2$ given price vector $p \in \Re^3_+ \setminus \{0^3\}$. By assumption, each

consumer $i \in I^2$ maximizes his utility function over his budget set. Under Assumptions 1 through 3, the solution of each consumer's maximization problem is unique and also satisfies the budget constraint with equality: $p^T d^i(p) = p^T w^i$, for each consumer $i \in I^2$ for every $p \in \Re^3_+ \setminus \{0^3\}$. The derived demand function of each consumer $i \in I^2$ denoted by $d^i(p) : \Re^3_+ \setminus \{0^3\} \to \Re^3$ is continuous. The excess demand function $z^i(p)$ of consumer $i \in I^2$ is defined as: $z^i(p) = d^i(p) - \omega^i$, for every $p \in \Re^{n+1}_+ \setminus \{0^{n+1}\}$. Then, $p^T z^i(p) = 0$ for every $p \in \Re^3_+ \setminus \{0^3\}$. The aggregate excess demand function $z : \Re^3_+ \setminus \{0^3\} \to \Re^3$ defined as: $z(p) \equiv \sum_{i \in I^2} z^i(p)$, for every $p \in \Re^3_+ \setminus \{0^3\}$ is a continuous function with the following properties:

Property 1: $p^T z(p) = 0, \ \forall p \in \Re^3_+ \setminus \{0^3\}$ (Strong form of Walras' law).

If every consumer is locally nonsatiated, then consumers spend all of their wealth on consumption. This results in $p^T z(p) = 0$ for every price p.[2]

[2]See Ellickson [1, 1993: 236-237] for the proof in detail.

Property 2: $z(p) > 0$, whenever $p_j = 0$ (desirability).

In this case, if some price is zero, the aggregate excess demand for the commodity is strictly positive. If all commodities are desirable and p^* is a Walrasian equilibrium, then $z(p^*) = 0$. This implies the equality of demand and supply of commodities at their equilibrium prices.

Property 3: $z(\mu \cdot p) = z(p), \ \forall scalars \, \mu > 0, \ \forall p \in \Re^3_+ \setminus \{0^3\}$ (homogeneity of degree zero in prices).

Note that homogeneity of degree zero in prices allows normalization of the price space to the n-dimensional unit simplex. Therefore the analysis of aggregate excess demand functions is restricted to the normalized price space formed by the unit simplex. The n-dimensional unit simplex S^n is defined by:

$$S^n \equiv \left\{ p \in \Re^{n+1}_+ \mid \sum_{j=1}^{n+1} p_j = 1 \right\}$$

In this example, $n = 2$. An equilibrium price vector is a price vector denoted by $p^* \in S^n$, where aggregate excess demand function $z(p^*) = 0^{n+1}$ holds for each commodity. It is well-known that such an equilibrium price vector p^* always exists on the n-dimensional unit simplex satisfying $z(p^*) = 0^{n+1}$.

In this model structure, the economy $\varepsilon = ((X^i, \preceq^i, \omega^i)_{i \in I^2})$ is such that for the consumption set $X^{i \in I^2} \subset \Re^3_+$. Each consumer's preference relations $\preceq^1$ and $\preceq^2$ are represented by utility functions $u^1 : X^1 \to \Re$ and $u^2 : X^2 \to \Re$, respectively, defined by

$$u^1(x^1) = (x^1_1)^{0.3}(x^1_2)^{0.3}(x^1_3)^{0.4}, \quad \forall x^1 \in \Re^2_+,$$
$$u^2(x^2) = (x^2_1)^{0.5}(x^2_2)^{0.2}(x^2_3)^{0.3}, \quad \forall x^2 \in \Re^2_+.$$

The vectors of initial endowments of the 3 commodities $j \in I^3$ of consumer $i \in I^2$ are given as:

$$\omega^1 = (\omega^1_1, \omega^1_2, \omega^1_3)^T = (40, 20, 30)^T,$$
$$\omega^2 = (\omega^2_1, \omega^2_2, \omega^2_3)^T = (20, 30, 40)^T,$$
$$\sum_{i=1}^{2} \omega^i = (60, 50, 70)^T.$$

Each consumer's income denoted by $Y^i \equiv p^T \omega^i, \ \forall i \in I^2$ is given as follows:

$$Y^1 = 40p_1 + 20p_2 + 30p_3,$$
$$Y^2 = 20p_1 + 30p_2 + 40p_3,$$

This example is taken from Washida [14, 2004, pp. 256-7]. As a result of each consumer's utility maximization, the

corresponding demand functions for consumer $i \in I^2$ are denoted by $d^i(p),\ \forall i \in I^2$ as:

$$d^1(p) = \left(\frac{0.3Y^1}{p_1}, \frac{0.3Y^1}{p_2}, \frac{0.4Y^1}{p_3}\right)^T, \quad \forall p \in \Re^3_+ \backslash \{0^3\},$$
$$d^2(p) = \left(\frac{0.5Y^2}{p_1}, \frac{0.2Y^2}{p_2}, \frac{0.3Y^2}{p_3}\right)^T, \quad \forall p \in \Re^3_+ \backslash \{0^3\}.$$

It is also easily verified that for every consumer $i \in I_2$, $d^i, \forall i \in I_2,$ is a continuous function. The Walrasian equilibrium price systems are given by the solutions $p^* \in \Re^3_+ \backslash \{0^3\}$ of the system of equations of the aggregate demand function $z(p) : \Re^3_+ \backslash \{0^3\} \to \Re^3$ as:

$$z(p) \equiv \sum_{i \in I^2} d^i(p) - \sum_{i \in I^2} \omega^i = 0.$$

Rewrite the above equation in matrix form as:

$$\begin{bmatrix} \frac{0.3Y^1}{p_1} \\ \frac{0.3Y^1}{p_2} \\ \frac{0.4Y^1}{p_3} \end{bmatrix} + \begin{bmatrix} \frac{0.5Y^2}{p_1} \\ \frac{0.2Y^2}{p_2} \\ \frac{0.3Y^2}{p_3} \end{bmatrix} - \begin{bmatrix} 60 \\ 50 \\ 70 \end{bmatrix} = 0.$$

A Walrasian equilibrium price system $p^* = (p_1^*, p_2^*, p_3^*)^T$ satisfying the condition $\sum_{j=1}^{3} p_j = 1$ is numerically computed (by the seven alternative approaches chosen in this paper) if and only if $p^* \in \Re^3_+ \backslash \{0^3\}$. The unique Walrasian equilibrium price vector p^* is given by:

$$p^* = (0.396791, 0.30187, 0.301333)^T.$$

The computational results for seven alternative approaches are presented in the Appendix. All assumptions made are satisfied by this economy ε.

3. Descriptions of the Seven Alternative Approaches

In this section, the seven algorithms applied to compute the benchmark model in Section 2 are briefly described as follows:

1. Scarf Approach (program: scarf3.c): This approach was first introduced by Scarf [8, 1967], and the exposition follows Shoven and Whalley [11, 1992:40-1.]. To find such a fixed point, the unit simplex is divided into a finite number of smaller simplices, each defined by N vertices that are each associated with a label. These labels are chosen from the set of integers that defined the dimensionality of the general equilibrium problem (if N is the dimensionality of the problem). The labels are chosen in such a way that if a simplex can be found whose vertices have a complete set of labels (from 1 to N) associated with them, then this implies that a close approximation to a general equilibrium must have been found. The operational program used in this paper was developed by Washida [14, 2004:256-7] and modified for the implementation.

2. Modified Kimbell-Harrison Approach (program: wsd-1.c): This approach was developed by Kimbell and Harrison [7, 1986:197-212.] and extended by Kawano [6, 2003:2-27.]. The crucial step for revising prices over iterations is to increase prices for all the excess demands and to lower all the excess supplies simultaneously. This Walrasian tatonnament process was simply formulated in the program as: $p_{n+1} = p_n\,(D/S)$, where $D :=$ excess demand and $S :=$ excess supply.

3. Joosten-Talman Approach (program: wsd-3.c): This approach was developed by Joosten and Talman [5, 1998:15-26.]. The purpose of the paper was to introduce a new globally convergent price adjustment process. Under the price adjustment process, only the prices of commodities having the highest excess demand are increased from their initial levels. At the same time, only the prices of

commodities having the lowest excess demand are decreased from their initial levels. This process was easily programmed for operational use.

4. Modified Tanaka-Kawano Approach (program: wsd-7.c): This approach was developed for solving a general equilibrium model by Tanaka and Kawano [12, 1996] and extended to solve the multi-dimensional case. The main feature is: if the sign of the same excess demand function changes, reduce the currently assigned incremental step length by half, if the absolute value of the excess demand function is increased, then reverse the direction of the incremental step length, if otherwise, reassign the same step length and continues the iteration.

5. Three Traditional Nonlinear Equation Solvers: the Bisection Method (program: wsd-8.c), the Secant Method (program: wsd-10.c), and **the Traub Method (program: wsd-11.c)**. An overview of the methods of the nonlinear solvers mentioned here has been briefly treated and restricted to the cases of univariate functions for illustrating each method.[3]

1) **Bisection Method**: This method is based on systematically reducing the interval of uncertainty by function comparison. Suppose that an initial interval $[a, b]$ has been specified in which $f(a)f(b) < 0$. We evaluate f at the midpoint of the interval and test its sign. If the function value is zero, the algorithm terminates. Otherwise, a new interval of uncertainty is produced by discarding the value of a or b, depending on whether $f(a)$ *or* $f(b)$ agrees in sign with f at the midpoint.

[3]The explanations were referred to Gill, Murray, and Wright [3, 1986: 83-87] and Togawa [13, 2000: 205-217] in detail.

2) **Secant Method**: This method is also called the method of linear interpolation. In Newton's method, f' is required at every iterate. f' may be very expensive, troublesome, or even impossible to compute. A different method is suggested by using the same idea of approximating f by a straight line. $f'(x_k)$ is replaced in the Newton formula by the finite-difference approximation $(f_k - f_{k-1})/(x_k - x_{k-1})$, where f_k denotes $f(x_k)$. The iteration is usually rapidly converged.

3) **Traub Method**: This is an improved version of Muller method which is based on approximating the function in the neighborhood of the root by a quadratic polynomial. This gives a much closer match to the actual curve, compared with the one approximating the function in the neighborhood of the root by a straight line. A second-degree polynomial is made to fit three points near a root and the proper zero of this quadratic formula is used as the improved estimate. This process is repeated until the iteration is converged.

4 Simulation Results

Among the seven alternative approaches starting with the same initial values, the best approach was the modified Kimbell-Harrison approach which achieved a high computational precision with only 20 iterations to obtain the converged values (see Program wsd-1.c in the Appendix). I applied this approach to the example drawn from Shoven and Whalley [10, 1974]. In Kawano [6, 2003: 2-27.], the number of iterations was 409 over the Scarf's algorithm, which Shoven and Whalley used to solve the same problem with 1653 iterations. The Joosten-Talman approach also achieved a high precision with just 30 iterations (see Program wsd-3.c in the Appendix). The modified Tanaka-Kawano approach achieved a high

precision, but after 176 iterations (see Program wsd-7.c in the Appendix). Among the traditional nonlinear solvers, the Bisection and the Traub methods performed well with high precisions, even though the number of inner loop iterations seems large (see Programs wsd-8.c and wsd-11.c in the Appendix). The Secant method performed very poorly, since it did not achieve a high precision with 5,000 iterations (see Program wsd-10.c in the Appendix). The Scarf approach performed very poorly and did not achieve a high precision with over 1 million iterations, even though its computational time was 0.142 seconds which is comparatively long (see Program scarf3.c in the Appendix). The other approaches took less than 0.001 second to converge. Even if the Scarf approach did poorly in this simulation, this discretized grid method could be used initially to search for approximately converged values over the entire *n*-dimensional unit simplex. Then, these approximately converged values could be fed into the other higher precision algorithms. These experiments were programmed in C-language, and conducted on the GCC version 4.0.1 compiler (Apple Computer, Inc.). The verified reliability of the simulation results in double precision (1.0e-15).

5. Conclusion

In order to see the empirical applicability of these seven alternative approaches, in this paper I conducted numerical simulations of the simple illustrative model and compared their performances. As a result of the simulation exercise, among them the best approach was the modified Kimbell-Harrison approach, which achieved a high computational precision with only 20 iterations to obtain the converged values. Since it is considered the most promising approach, I applied the method to the example drawn from Shoven and Whalley [10, 1974], and it successfully computed the model.

(Received : November30,2012, Accepted : January30,2013)

References

[1] Ellickson, Bryan (1993) *Competitive Equilibrium: Theory and Applications.* Cambridge, UK: Cambridge university Press.

[2] Dixon, Peter H., Brian R. Parmenter, and Alan A. Powell (1992) *Notes and Problems in Applied General Equilibrium Economics*. Amsterdam: North-Holland.

[3] Gill, Philip E., Walter Murray, and Margaret H. Wright (1986) *Practical Optimization.* London, UK: Elsevier Academic Press.

[4] Johansen, L. (1960) *A Multisectoral Study of Economic Growth*. Amsterdam: North-Holland (Second Edition 1974).

[5] Joosten, R., and Dolf Talman (1998) "A Globally Convergent Price Adjustment Process for Exchange Economies." *Journal of Mathematical Economics* 29 (July): pp. 15-26.

[6] Kawano, H (2003) "A Simultaneous Multi-Factor Price Revision for Solving a Numerical General Equilibrium Model." *Aomori Public College Journal of Management & Economics* Vol. 8, No. 2, pp. 2-27.

[7] Kimbell, L. J., and G. W. Harrison (1986) "On the Solution of General Equilibrium Models." *Economic Modelling* 3 (July): pp. 197-212.

[8] Scarf, H.E. (1967) On the Computation of Equilibrium Prices. In *Ten Economic Essays in the Tradition of Irving Fisher*. New York: Wiley.

[9] Scarf, H.E. (with T. Hansen) (1973) *The Computation of Economic Equilibria.* New Haven: Yale University Press.

[10] Shoven, J. B., and J. Whalley (1974) "On the Computation of Competitive Equilibrium on International Markets with Tariffs." *Journal of International Economics* 4: pp. 341-354.

[11] Shoven, J. B., and J. Whalley (1992) *Applying General Equilibrium.* Cambridge: Cambridge University Press.

[12] Tanaka, H., and H. Kawano (1996) A Simple Fixed Point *Algorithm for a Static General Equilibrium Model.* Discussion Paper Series No.2, Aomori Public College.

[13] Togawa, Hayato (2000) *Kagakugijyutu Keisan Handobukku (Handbook for Scientifc Calculations)*.Tokyo: Saiensu Sya.

[14] Washida, T (2004) *Kankyo Seisaku to Ippankinko (Environmental Policy and General Equilibrium)*. Tokyo: Keisoshobo.

APPENDIX

A Computational Results

═══ The converged values ═══

PROGRAM: scarf3.c
ALGORITM: Scarf

/// 1.*Computational time & Iterations* ///

1－1) Initial values: p[0]=0.999998
p[1]=0.000001

1－2) Iteration for general equilibrium loop: No.= 1206418

1－3) Computational time: 0.142 seconds have passed.

/// 2. *Commodity prices Prices[i]* ///

2－1) p[0]= 0.396790000000000
2－2) p[1]= 0.301877000000000
2－3) p[2]= 0.301333000000000

/// 3. *Excess demands for commodity markets rho L[k],rho K[k]* ///

3－1) rho[0]= 0.000136092139428
3－2) rho[1]=－0.000082815186317
3－3) rho[2]=－0.000096239044510

═══ The converged values ═══

PROGRAM: wsd－1.c
ALGORITHM: Modified Kimbell-Harrison

/// *1. Computational time & Iterations* ///

1－1) Initial values: p[0]=0.999998
p[1]=0.000001

1－2) Iteration for general equilibrium loop: No.= 20
1－3) Computational time: 0.000 seconds have passed.

/// 2. Commodity prices p[i] ///

2－1) p[0]= 0.396790862115855
2－2) p[1]= 0.301876529779712
2－3) p[2]= 0.301332608104433

/// 3. Excess demands for commodity markets rho[i] ///

3－1) rho[0]= 0.000000000000000
3－2) rho[1]=－0.000000000000007
3－3) rho[2]= 0.000000000000014

═══ The converged values ═══

PROGRAM: wsd－3.c
ALGORITHM: Joosten..Talman

/// 1. Computational time & Iterations ///

1－1) Initial values: p[0]=0.999998
p[1]=0.000001

1－2) Iteration for general equilibrium loop: No.= 30
1－3) Computational time: 0.000 seconds have passed.

/// 2. Commodity prices p[i] ///

2－1) p[0]= 0.396790862115855
2－2) p[1]= 0.301876529779712
2－3) p[2]= 0.301332608104433

/// 3. Excess demands for commodity markets rho[i] ///

3－1) rho[0]= 0.000000000000000
3－2) rho[1]=－0.000000000000007
3－3) rho[2]= 0.000000000000014

═══ The converged values ═══

PROGRAM: wsd－7.c
ALGORITHM: Modified Tanaka..Kawano

/// 1. Computational time & Iterations ///

1－1) Initial values: p[0]=0.999998
p[1]=0.000001
1－2) Iteration for general equilibrium loop: No.= 176
1－3) Computational time: 0.000 seconds have passed.

/// 2. Commodity prices p[i] ///

2－1) p[0]= 0.396790862115855
2－2) p[1]= 0.301876529779712
2－3) p[2]= 0.301332608104433

/// 3. Excess demands for commodity markets rho[i] ///

3－1) rho[0]= 0.000000000000028
3－2) rho[1]= 0.000000000000000
3－3) rho[2]=－0.000000000000028

═══ The converged values ═══

PROGRAM: wsd－8.c
ALGORITHM: bisection method

/// 1. Computational time & Iterations ///

1－1) Initial values: p[1]=0.000001
1－2) Iteration for general equilibrium loop: No.= 7
Bisection inner loop1: No.= 32767
Bisection inner loop2: No.=1635511312

1－3) Computational time: 0.000 seconds have passed.

/// 2. Commodity prices p[i] ///

2－1) p[0]= 0.396790862115855
2－2) p[1]= 0.301876529779712
2－3) p[2]= 0.301332608104433

/// 3. Excess demands for commodity markets rho[i] ///

3－1) rho[0]= 0.000000000000000
3－2) rho[1]=－0.000000000000007
3－3) rho[2]= 0.000000000000014

═══ The converged values ═══

PROGRAM: wsd－10.c
ALGORITHM: secant method

/// 1. Computational time & Iterations ///

1－1) Initial values: p[1]=0.000001
1－2) Iteration for general equilibrium loop: No.=5000
1－3) Computational time: 0.010 seconds have passed.

/// 2. Commodity prices p[i] ///

2－1) p[0]= 0.396788992008012
2－2) p[1]= 0.301876563781672
2－3) p[2]= 0.301334444210315

/// 3. Excess demands for commodity markets rho[i] ///

3－1) rho[0]= 0.000315092435322

3－2) rho[1]= 0.000000000000000
3－3) rho[2]=－0.000414905140133

═══ The converged values ═══

PROGRAM: wsd－11.c
ALGORITHM: Traub method

/// 1. Computational time & Iterations ///
1－1) Initial values: p[1]=0.000001
1－2) Iteration for general equilibrium loop: No.= 7
1st inner loop1: No.= 32767
1st inner loop2: No.= 1678281744
1－3) Computational time: 0.000 seconds have passed.

/// 2. Commodity prices p[i] ///

2－1) p[0]= 0.396790862115855
2－2) p[1]= 0.301876529779712
2－3) p[2]= 0.301332608104433

/// 3. Excess demands for commodity markets rho[i] ///

3－1) rho[0]= 0.000000000000000
3－2) rho[1]=－0.000000000000007
3－3) rho[2]= 0.000000000000014

═══ The end of the output flie ═══